REMARKS

The Office Action mailed December 31, 2008, considered and rejected claims 1-4, 6-16 and 18. Claim 16 was objected to because of informalities. Claims 1-4, 6-16, and 18 were rejected under 35 U.S.C. § 103(a) as being unpatentable over *Baisley* et al., (U.S. Patent No. 6,106,574) in view of *Haikin* (U.S. Patent No. 6,757,893), *Murakami* (U.S. Publ. No. 2003/0167423) and *Boxall* et al. (U.S. (U.S, Patemt No. 6,263,451).[1]

By this paper, no claims are added, amended, or cancelled. Accordingly, following entry of this paper, claims 1-4, 6-16 and 18 remain pending, of which claims 1 and 11 are the only independent claims at issue.

As reflected above, Applicants claims relate generally to associating original source code with binary code for debugging purposes. As reflected in claim 1, for example, a computer-readable storage medium has computer-executable instructions that are executable by a processor to implement a method for associating original source code with binary code for debugging the binary code. The method itself includes storing a source code file on a server, the source code file including source code and being associated with a version. The source code is compiled into a binary file and while being compiled various information is extracted. That information includes: a location of the source code file, the version associated with the source code file, a name of the server, a port of the server at which the server may be accessed to access the source code, a path to the source code, and a numeric value that indicates a version number of the source code. The extracted information is stored in a debug file associated with the binary file. After compiling the source code file, an instruction for a debugger to debug the binary file is received. Thereafter, the extracted information in the debug file is used to locate the source code file and associated it with the binary file. The binary file is then debugged with full source code support by correlating lines of source code with binary instructions in the binary file. Claim recites a similar system with various additional limitations not found within claim 1.

Applicant respectfully submits that while *Baisley, Haiken, Murakami,* and *Boxall* generally relate to compiling source code and/or debugging code, they fail—whether cited alone or in combination—to disclose or reasonably support the pending claims as recited in their

[1] Although the prior art status of the cited art is not being challenged at this time, Applicant reserves the right to challenge the prior art status of the cited art at any appropriate time, should it arise. Accordingly, any arguments and amendments made herein should not be construed as acquiescing to any prior art status of the cited art.

entireties. For example, among other things, the cited art fails to disclose or reasonably support extracting *at the time of compiling* a port number identifying a port of the server on which the source code may be accessed.

In particular, the Office has acknowledged the deficiencies of *Baisley, Haiken* and *Murakami* in this regard as none reasonably support the use of a port number in a compiling and/or debugging code. For this purpose, the Office uses the *Boxall* reference. In particular, *Boxall* relates to a system for remote debugging of client/server applications. The system includes a kicker program and a debugging engine stored on a server, and a debugging UI on a client machine. The kicker starts the debugging engine, which in turn starts the debugging UI on the client. (*Abstract*). In operation, the kicker program is referred to as debug.dll and is supplied with the debugger program. The debug.dll is loaded on the server machine and, once loaded, calls various entry points. One entry point is the _DBG_Start(parms) entry point that includes, among other things, an optional port number that identifies the port on which the debugger UI is listening. (Col. 6, ll. 50-52).

As noted in *Boxall*, the port number "identifies the port on which the debugger UI on the client is listening. The port itself is received from the client. The client provides the information after proxy code on the client "looks for environment settings for the network address or IP and port number where the debugger UI 19 is to be started." (Col. 7, ll. 35-40). As further clarified later, the port number is specified on a command line or by reading the port number from a services file, or within an environment variable (e.g., DEBUG_UICLIENT). (Col. 8, ll. 48-56).

Notably, *Boxall* thus teaches that the client provides the server .dll with the port number on which the client debugger UI operates, by looking at environment settings. This is in direct contrast with the pending claims which do not look to environment settings, but look instead to the specific debug file associated with a source file. In other words, Applicant's claims recite that while source code is being compiled, a port of the server at which the server may be accessed to access the source code is extracted; whereas *Boxall* teaches that the only port number is one associated with the client debugger UI, is stored as an environment variable and is apparently associated with the debugger irrespective of any particular source file. Indeed, nothing in *Boxall* in any way appears to indicate that the port number is associated with the server, let alone the port on the server at which the source code may be accessed. Moreover, there is no reason to extract the port number during compiling (as recited in the claims) inasmuch

as the client debugger UI already specifies a port as an environmental variable.

Thus, the cited combination fails to provide reasonably support for extracting a port number of the server while compiling the source code, and instead only teaches use of a port number by a client debugger UI as stored in an environmental variable, and apparently without regard to any particular server or source code file. Inasmuch as the port number in *Boxall* is therefore without regard to the server itself or the source code, there is no reasonable support for extracting the same from the source file during compiling.

With regard to claim 11, Applicant respectfully submits that the Office has failed to assert, much less provide, a *prima facie* case of obviousness. As reflected on page 8 of the Office Action, the full basis of rejection for claim 11 is based on reference to the method of claim 1. (i.e., "Haikin also discloses a system []for implementing the above method). The Office gives no consideration to the various elements in claim 11 that are not recited within claim 1. For example, claim 11 recites:

- an extractor operating in parallel with the compiler; and
- plurality of numeric values each indicating a version number of a corresponding source code file.

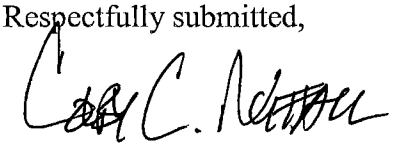
At no time does the Office consider such elements, particularly "multiple numeric values" that are extracted by an extractor "in parallel with the compiler" as recited in combination with the other claim elements. Indeed, the cited art fails to even disclose a single numeric value indicating a version number of a corresponding source code file, let alone that such is extracted during compiling. Specifically, the Office acknowledges that *Baisley* fails to disclose a numeric number indicating a version number of the source code. The Office then recites *Haiken* which keeps historical tracking by providing access to each version of source code. It appears, however, that *Haikin* operates by storing new files for each version, and therefore includes different paths to each file version, such that there is no disclosure of a numeric version number or extraction of a version number. Indeed, the Office doesn't even assert that *Haikin* discloses extracting a numeric number indicative of a version number of source code, does note that *Baisley* is deficient in the disclosure thereof, and provides no consideration of versions with respect to *Murakami* and *Boxall*. (*See* Office Action, pp. 5-8).

In view of the foregoing, Applicant respectfully submits that the other rejections to the claims are now moot and do not, therefore, need to be addressed individually at this time. It will be appreciated, however, that this should not be construed as Applicant acquiescing to any of the purported teachings or assertions made in the last action regarding the cited art or the pending application, including any official notice. Instead, Applicant reserves the right to challenge any of the purported teachings or assertions made in the last action at any appropriate time in the future, should the need arise. Furthermore, to the extent that the Examiner has relied on any Official Notice, explicitly or implicitly, Applicant specifically requests that the Examiner provide references supporting the teachings officially noticed, as well as the required motivation or suggestion to combine the relied upon notice with the other art of record.

In the event that the Examiner finds remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney at (801) 533-9800.

Dated this 1st day of June, 2009.

Respectfully submitted,



RICK D. NYDEGGER
Registration No. 28,651
COLBY C. NUTTALL
Registration No. 58,146
Attorneys for Applicant
Customer No. 047973

RDN:CCN:gd